The Unity Security Initiative

Paul England & Ken Ray
Windows Security

Outline

- How we got here
- What we are trying to do
- Security characteristics of the MS hypervisor
  - Defining and controlling the TCB
- Guests and Security
  - Unity building blocks
  - Converting Windows into a high assurance OS
- Conclusions

Outline

- How we got here
- What we are trying to do
- Security characteristics of the MS hypervisor
  - Defining and controlling the TCB
- Guests and Security
  - Unity building blocks
  - Converting Windows into a high assurance OS
- Conclusions

How we got here: NGSCB -> Unity

- We have accelerated support for hosting arbitrary OSs
- Original plan was
  - V1: Isolation kernel
    - Runs "primary"
    - Runs one or more high assurance OS guests
    - Results in many notable simplifications
  - V2
    - Adds better virtualization support to the machine monitor
    - Adds richer VM services
- Unity
  - Skips v1
    - Cuts the high assurance OS development
      - Replaces it with Windows variants

Outline

- How we got here
- **What we are trying to do**
- Security characteristics of the MS hypervisor
  - Defining and controlling the TCB
- A work in progress…
  - Unity building blocks
  - Converting Windows into a high assurance OS

Unity Security Scenarios

- Multiple Windows instances with different security configurations
  - Locked down
  - Open
  - Disposable (snapshot, rollback)
- High assurance guest
- Protection against offline attacks
- Challenges
  - Avoiding primary guest security dependencies
  - Making Windows secure

Outline

- How we got here
- What we are trying to do
- **Security characteristics of the MS hypervisor**
  - Defining and controlling the TCB
- Guests and Security
  - Unity building blocks
  - Converting Windows into a high assurance OS

Security Characteristics of the MS Hypervisor

- The MS Hypervisor owns / controls
  - CPU
  - Memory
  - IO address space
  - DMA protection
  - IPC primitives
  - Crypto processor (TPM)
- All other devices are exported to guests
  - Most to the primary guest

MS Hypervisor Security Challenges

- V1 hypervisor services are insufficient for an OS
  - Guests must use peer services for most IO
- E.g. the hypervisor is loaded from the disk
  - The disk is owned by a guest
  - i.e. the hypervisor itself has a security dependency on one of its guests
- So how big is the TCB?
  - All of Windows ~ 100 MLOC ?
  - And since the primary has the best performance (particularly graphics), this is will be the most adversarial environment on the machine

Possible Solutions to the TCB Problem

- Lock down the primary
  - Largely indistinguishable from a hosted VMM
- Move more drivers into Hv
  - Open driver model
  - Need to replicate much of ntoskrnl
- Move important drivers / services to other guests
  - E.g. separate MS from 3rd party drivers
  - This is the long term vision
    - Does it make sense?
  - This might be a decade long endeavor
- Use crypto
  - Encryption, HMAC, etc.
  - The TPM

Crypto and the TPM

- Where to store keys?
- TPM – Trusted Platform Module
- Defined by the Trusted Computing Group
  - Primary historical contributors
    - Microsoft, Intel, HP, IBM, Compaq
  - Now
    - AMD, Infineon, Atmel, ST Micro, Wave Systems, Verisign, HP, Microsoft, IBM, Intel

Primary TPM Features

- Measures software booting on the platform
- Protects keys
- Implements crypto algorithms
- A half a dozen other feature areas

TPM Software Authentication

- Starting a hypervisor
  - The processor / TPM
    - Performs a lightweight reset
    - "Measures the hypervisor" as it initializes
    - Starts executing the processor in a defined mode
  - Independent of how Hv got into memory
- The TPM then
  - Remembers the "hash of Hv"
  - Provides security services (e g keys) to hypervisors gated by the Hv hash

Using the TPM

- Hv (or a privileged guest) can
  - Store keys inaccessible to unauthorized Hvs
  - Provide similar (virtualized) services to its guests
- Practically
  - Each OS can store keys only accessible to authorized SW
  - No security dependency on other guests for this service
  - Crypto provides
    - Confidentiality
    - Integrity protection
    - Does not protect against DOS (deletion)

Unity Security Characteristics

- Memory
  - Wholesaled by primary
  - Access protected by Hv following donation
    - Primary can starve Hv
- CPU
  - TCB smaller if CPU instruction set is virtualizable
    - e.g. Intel Lagrande Technology
  - Scheduled by primary
    - Primary can starve Hv
- IO Devices
  - Nearly all owned by primary
  - Implications/remediation in following slides

Network Security

- **Baseline Hv security characteristics**
  - Net card owned by primary
  - Net stack in primary routes packets to/from other guests
  - Primary can examine / modify all packets
- **Guest security remediation**
  - IPSEC, VPN, SSL, etc tunnels
  - TPM can help with key storage

Disk Security

- Baseline Hv characteristics
  - Disk owned by primary
  - Guests use virtual volumes managed by primary
  - Hypervisor executable loaded from disk
  - All other guests loaded from disk
- Guest security remediation
  - Authenticated load of Hv
  - Guest can use TPM
  - Encrypt and/or integrity protect guest data
    - File, volume, disk, level
    - Guest can do this (but EFS needs to be fixed)
    - We will provide a system service
      - Crypto protected virtual disk

Input security

- Baseline Hv characteristics
  - USB host controller / i8042 owned by primary
  - Primary HID stack inspects / routes user input
  - Primary spyware can inspect / modify all input
    - Including local Hv administration
- Unity remediation
  - Modify USB HC to allow standard driver in the TCB
    - Primary can still starve in v1

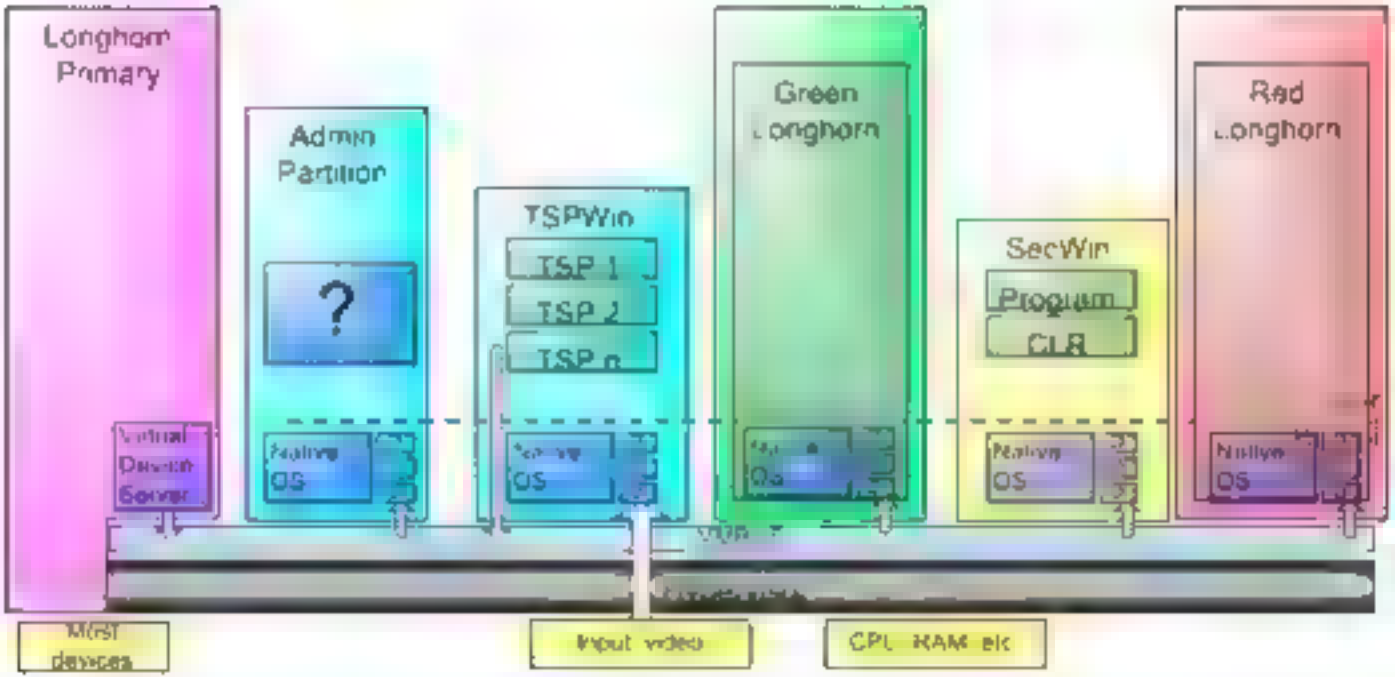
Video Output Security

- Biggest challenge
- Baseline Hv characteristics
  - Video card owned by primary
  - Primary / driver sees and can modify all video output
- Guest security remediation
  - Conceptually: second video card and driver
  - Current State: Non accelerated 2D surface
  - Probably need to wait until GPU innovation slows down to do much better

Outline

- Guests and Security
  - Unity building blocks

Unity Partition Model



Unity Building Blocks - I

- **Hypervisor**
  - Manages CPU, memory, APIC, DEV,
- **Primary (Longhorn)**
  - Manages nearly all devices
  - Starts Hv
  - Exposes virtual device services to peers

Unity Building Blocks - II

- MinWin / Componentization
  - Base OS build products
- TspWin - <1MLOC
  - Ntoskrnl + hal, + drivers + one or two processes
  - Part of TCB for high security guests
  - Drivers for "trusted hardware"
  - Virtual device servers
  - Authenticated by Hv
  - "building block" for post LH OSs
- AdminWin – 10 MLOC
  - Longhorn with most services removed
  - UI administration
  - Network administration

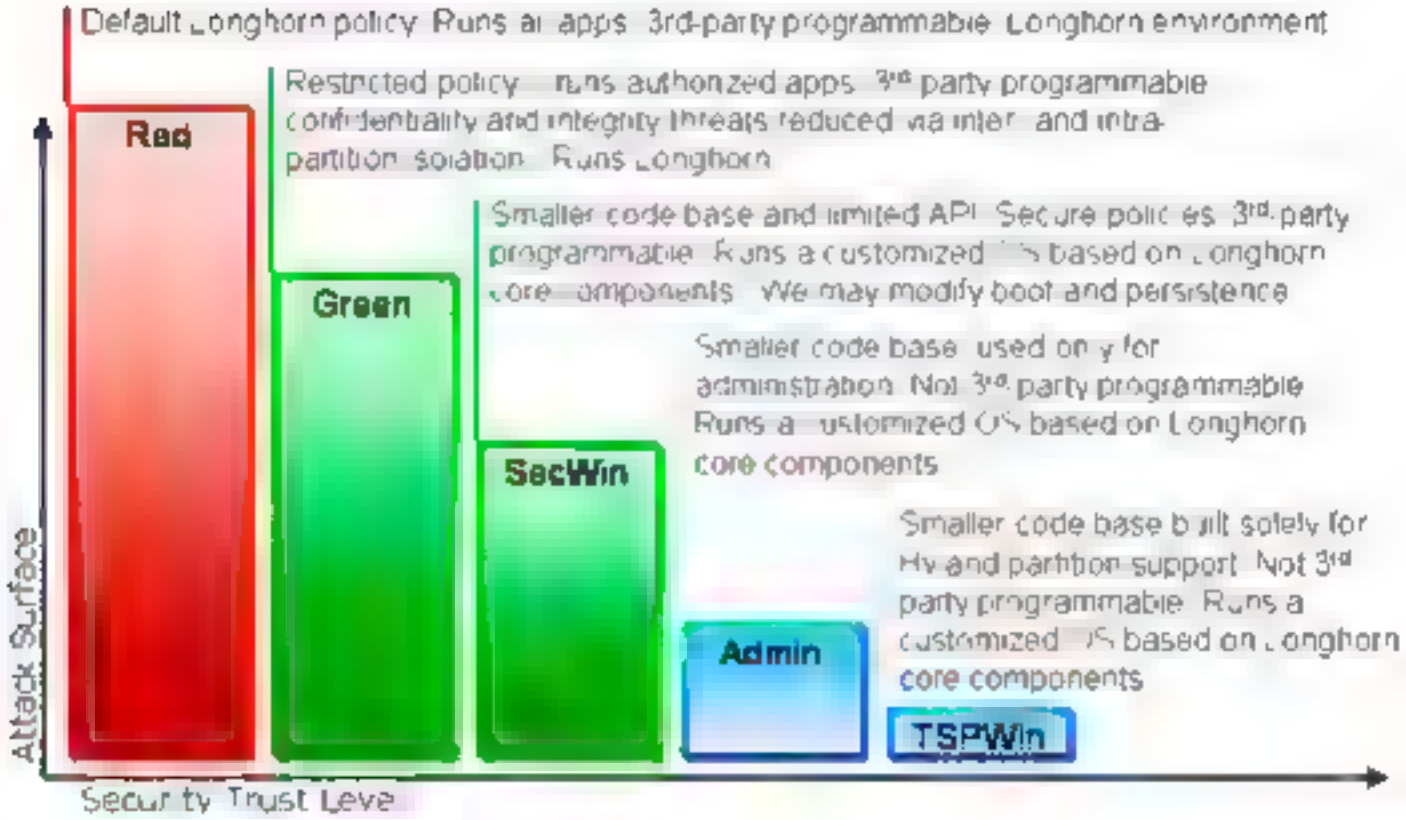
Unity Building Blocks - III

- SecWin – 12 MLOC
  - Longhorn with most services removed
  - High assurance programming environment
  - CLR

Unity Building Blocks – IV

- Longhorn Green ~ 100MLOC
  - Longhorn configured for security
- Longhorn Red ~ 100MLOC?
  - Longhorn loosely configured
- Value proposition (not yet tested)
  - It is possible to lock down XP, LH today
    - This is not done because it gets in the way
  - We can provide 2 or 3 partitions these can be configured differently
    - One for home banking
    - One for surfing the web
    - One managed by the IT department
    - One managed by your kids
  - Exploit rate of "green" will be lower than "red"
  - Guest exploits are contained with the guest
  - Usability is key

Partition Taxonomy



Outline

- Converting Windows into a high assurance OS

Security Challenges of Windows

- The NGSCB "high assurance guest" was going to be a few 10s of KLOC
- Current smaller useful windows configurations are >10MLOC
- This is not necessarily a bad tradeoff
  - But it means work

Making Windows Secure

- Internal Guest Protections
  - Leave stuff out
  - Configure the OS for security
- External Protections
  - Control what gets in
  - Control what gets out

Leaving stuff out of Windows

- Componentization
  - Effort to make Windows more modular
    - Reduce inter-module dependencies
  - MinWin is a Windows base build target
  - Other teams are fixing dependencies
  - We're driving some of this
- But this is hard problem…

Windows Security Configuration

- Relevant features in Longhorn
  - OS integrity
    - Secure boot
    - Code integrity (driver loading, etc.)
  - Network protection
    - Personal firewall
    - IPSEC "require"
    - VPNs
  - API bug remediation
    - Software restriction policies (SRP)
      - Administrator control of authorized programs, DLLs
  - Remote administration
    - Restricted token local users
  - MAKO (behavior blocking)
  - Virus checkers
  - LUA
  - Other springboard features
- No need for third party drivers in a VM
- Hope to provide tools, security configuration skeletons, etc.

External Protection

- **Peer authentication**
- **Discretionary access control for shared objects**
- **Mandatory Access Control (MAC)?**
  - **Not planned for v1**
- **The network is the interesting resource**
  - **Hypervisor "MAC firewall"?**
    - Inbound firewall is likely same code as Longhorn
      - Some security benefit in an external firewall
      - Inbound attacks will probably be exploiting the same vulnerabilities
      - Permits different hypervisor and OS administrators
    - Outbound
      - Hv admin can limit hosts, IPSEC/VPN tunnels, etc. that guest can connect to
      - Limits data flow following guest exploit
      - Threat model? Net attacker? Local user?
  - **Network firewall challenges**
    - Tunneling protocols (email)
    - How far up the network stack does the firewall have to go?

SecWin (beyond green)

- Longhorn green is still fragile
  - Once exploited it's exploited forever
- Remediation
  - Out of proc virus checkers
  - Better protection of the OS
    - One instance of the OS per application
      - E.g. implementation: read only system drive
    - Only managed code
    - API restrictions

Building Longhorn Red

- Configuring for low security is the easy part…
- What else?
  - Make it easy to rebuild?
  - Make it stateless / disposable?
  - Out of proc virus checkers?
- Clear use model is key

Red / Green Challenges

- **User experience**
  - User model
    - User has to change context?
    - System changes context on behalf of user?
      - Data driven?
  - Data sharing
    - File system?
    - Clipboard?
  - UI
    - Integrated desktop?
    - Fast user switching?
    - Recursive desktops?
  - Single sign on?
- **Administration**
  - Installing applications?
  - Upgrades?
- **What is an OS?**
  - How separate should the Windows guests be?